# Segmentation of tree seedling point clouds into elementary units

Franck Hétroy-Wheeler, Eric Casella, Dobrina Boltcheva

To appear in the *International Journal of Remote Sensing*
Vol. 00, No. 00, Month 20XX, 1–27

# Segmentation of tree seedling point clouds into elementary units

Franck Hétroy-Wheeler[a*], Eric Casella[b*] and Dobrina Boltcheva[c]

[a] *Univ. Grenoble Alpes & Inria, Laboratoire Jean Kuntzmann, 655 avenue de l'Europe, F-38334 Saint Ismier cedex, France*
[b] *Centre for Sustainable Forestry and Climate Change, Forest research, Farnham, Surrey, GU10 4LH, United Kingdom*
[c] *Univ. Lorraine & Inria, LORIA, Nancy, France*

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

This paper describes a new semi-automatic method to cluster TLS data into meaningful sets of points to extract plant components. The approach is designed for small plants with distinguishable branches and leaves, such as tree seedlings. It first creates a graph by connecting each point to its most relevant neighbours, then embeds the graph into a spectral space, and finally segments the embedding into clusters of points. The process can then be iterated on each cluster separately. The main idea underlying the approach is that the spectral embedding of the graph aligns the points along the shape's principal directions. A quantitative evaluation of the segmentation accuracy, as well as of leaf area estimates, is provided on a poplar seedling mock-up. It shows that the segmentation is robust with false positive and false negative rates around 1%. Qualitative results on four contrasting plant species with three different scan resolution levels each are also shown.

**Keywords:** terrestrial laser scanning; tLiDAR; 3D point cloud; segmentation; spectral clustering; tree seedling

## 1. Introduction

Functional-structural plant models describe a plant as a collection of interconnected elementary units (internode, petiole, leaf-blade, see Godin et al. (1999)). Their goal is to help biologists understand the relationships between the plant structure and the biological and physical mechanisms underlying the plant growth (Godin and Sinoquet (2005)). These models require an *in situ* validation on real plants, that can be done by measuring the three-dimensional (3D) characteristics of vegetation. Similarly the growing field of plant phenomics, which is concerned with the discovery and analysis of complex plant traits (Furbank and Tester (2011); International Plant Phenotyping Network (2016)), requires the measurement of individual quantitative parameters such as leaf characteristics.

Destructive measurements have long been used but are time consuming and expensive. As a consequence, various kinds of sensors are being investigated

*Corresponding authors. Emails: Franck.Hetroy@grenoble-inp.fr; Eric.Casella@forestry.gsi.gov.uk

for non-destructive and non-invasive plant metrology. For example, the use of different imaging techniques has been proposed for plant phenotyping, see Li et al. (2014) for a review. The most popular imaging techniques are based on single-lens cameras (Quan et al. (2006); Paproki et al. (2012)), time-of-flight cameras (Chéné et al. (2012); Alenyà et al. (2013); Chaivivatrakul et al. (2014); Xia et al. (2015)) or multi-view stereo imaging systems (Golbach et al. (2015); Lou et al. (2015); Rose et al. (2015)). All these methods allow one to reconstruct and measure single leaves, although some of them require manual interaction (Quan et al. (2006); Golbach et al. (2015); Rose et al. (2015)) or prior knowledge of the plant (Chéné et al. (2012); Chaivivatrakul et al. (2014)).

Terrestrial laser scanning (TLS), a remote sensing technique, has become an increasingly popular technique to measure vegetation from grass to forest plant species, see e.g. Dassot et al. (2011); Lin (2015) for recent reviews. Compared to imaging techniques, TLS provides direct accurate 3D measurements. It has also proved to be more robust to diverse environments, in particular to changing lighting conditions (Li et al. (2014); Lin (2015)). Thus, LiDAR seems more adapted to greenhouse and field conditions (Tilly et al. (2012); Lin (2015)).

TLS generates unstructured sets of points where its laser beam is incident and reflected. Thus it gives a raw sketch of the spatial distribution of plant elements in 3D, but it lacks explicit and essential information on their shape and connectivity. The points need to be clustered into geometrically meaningful sets for further analysis and dendrometric measurements. For example, leaf-blade points need to be separated from petiole and internode points to assess leaf areas.

In this paper, we segment TLS data of small plants or tree seedling scans into their elementary units: internodes, petioles and leaf-blades. Our method only considers the 3D positions of the points (no intensity value or normal estimate is required). As a consequence, it can be applied to sets of 3D points generated by other techniques than TLS such as time-of-flight cameras. We focus on accurate segmentation so that individual elementary unit characteristics such as leaf area are estimated as accurately as possible.

### 1.1. *Related work*

Segmentation of 3D data is critical for many applications in science. Research has considered the segmentation of point clouds into basic geometric primitives (planes, cylinders, spheres, etc.) for various purposes, such as building or city modelling (see Haala and Kada (2010) for a survey), reverse engineering of mechanical objects (e.g. Bey et al. (2011); Li et al. (2011)), or background subtraction, see Nguyen and Le (2013) for a recent overview. These approaches are designed for man-made objects which can be almost completely decomposed into uniform geometric shapes. Yet since stems and leaves are not exactly cylindrical and planar shapes, their efficiency to robustly segment plants is questionable.

The problem we are interested in, that is the segmentation of plants into their elementary units, has been partially tackled in the literature. Recovering the branching structure of a plant or a leaf-on tree is a specific issue that has been addressed in order to estimate various wood parameters by e.g., Bayer et al. (2013); Belton et al. (2013). Some authors additionally propose to reconstruct

the foliage, using some heuristics to position them, for example to create visually pleasing virtual 3D models of trees from TLS point clouds (e.g., Xu et al. (2007); Livny et al. (2010)) or to derive global characteristics such as total leaf and wood areas (Côté et al. (2009)). The number of leaves as well as their individual location and shape are plausible but do not correspond to the actual tree.

Other works segment a plant into two clusters only, one for the stems and one for the leaves, mostly by classifying points according to local geometric features (Belton et al. (2013); Paulus et al. (2013, 2014); Wahabzada et al. (2015)). Other approaches use geometric distance information (Tao et al. (2015)) or intensity information (Douglas et al. (2015)). Deriving a full segmentation of the plant from such a classification is possible in some cases, using prior knowledge about the plant or its organs (Paulus et al. (2014)). However this is not straightforward in general, especially when leaves are almost overlapping.

Geometric segmentation of a plant into elementary units has been proposed using either plant-specific prior knowledge (e.g., Kaminuma et al. (2004)) or a tedious interactive procedure (Dornbusch et al. (2007); Hosoi et al. (2011); Paulus et al. (2014)). In contrast, our approach only requires a minimum user interaction, and no prior knowledge. It is thus applicable to any species.

Yin et al. (2015) have recently proposed a destructive approach to accurately segment and reconstruct a pot plant. Their approach requires laser scanning the whole plant, manually cutting the plant leaves and laser scanning each leaf individually. This is time-consuming and obviously does not allow for tracking changes in the plant traits over time.

A approach analogous to that used here has recently been published by Lou et al. (2015), although they work on point cloud data generated using a multi-view stereo imaging system. The methodology is similar: a graph is first constructed to build neighbourhood relationships between the plant's points, then a spectral clustering approach is performed on this graph. However, we used more advanced graph construction and clustering techniques, as detailed in Sections 2.1.2 and 2.1.4.

### 1.2. *Approach*

The main contributions of this paper are:

(1) a semi-automatic approach to accurately segment a TLS 3D point cloud of a tree seedling into meaningful clusters of points (Section 2.1). More specifically, each cluster gathers points of an elementary unit of the tree seedling.The approach is global in the sense that leaves are not segmented first from branches, and robust to non uniform density in the point cloud;
(2) the assessment of the method robustness by quantitatively evaluating clustering results and individual leaf-blade areas on computer-generated scans from a plant mock-up (Sections 3.2.1 and 3.2.2).

Qualitative validation on real plant scans and parameter sensitivity analysis are also given (Sections 3.1 and 3.2.3, respectively).

## 2. Materials and methods

The segmentation algorithm underlying the approach is first explained in detail in Section 2.1. Reference data used in the experiments is then described in Section 2.2. The method used for statistical analysis of leaf area estimates is explained in Section 2.3.

### 2.1. *Point cloud segmentation algorithm*

In this section, the algorithm that analyses data collected from the TLS is described. The input point cloud data is merely a 3D location of points with no additional information. The algorithm is designed to cluster points into subsets corresponding to the plant elementary units. This algorithm is a three-stage process (Figure 1).

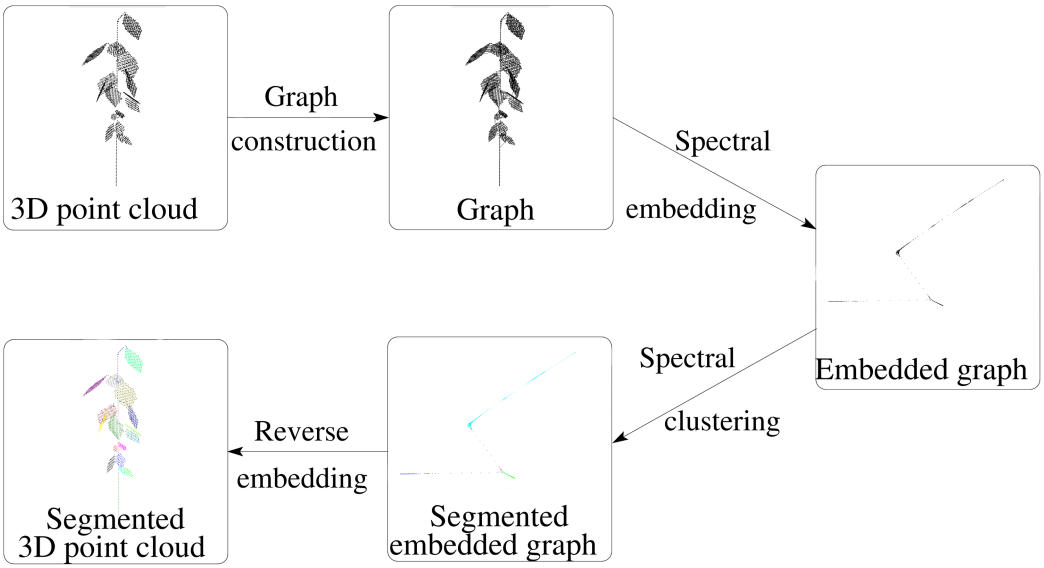



Figure 1. Pipeline of the segmentation method.

The first stage, called Graph Construction, finds neighbouring points for each point of the raw TLS data (see Section 2.1.2 for details). The second stage, called Spectral Embedding, finds the major intrinsic plant directions, i.e. the main directions of each elementary unit (see Section 2.1.3). This allows us to define the distances between neighbouring points according to the intrinsic plant directions, rather than the usual Euclidean distance. For example, the distance between two points sampled on a leaf-blade with an ellipsoid shape corresponds to the distance between their projections on the leaf's midrib (ellipsoid's main axis). Thus, two points on both sides of the ellipsoid's main axis but with similar projections will appear close to each other. As a consequence, this stage of the algorithm transforms the raw TLS data into a cloud of points aligned along principal plant axes (see Figure 2). Finally, the third stage, called Spectral Clustering, uses the computed neighbouring relationships to decompose the shape into subsets of points according to the principal plant axes. All points in a subset are given the same label (in our experiments, a colour), and points in different subsets have different labels. During this stage, each elementary unit is thus split from the one it originates. For example, a leaf-blade is separated from its petiole (see Section 2.1.4). Since each

point in the embedding space corresponds to a point in the Euclidean space, the segmentation of the input TLS data is automatically found by giving to each point the label of its associated point in the embedding space (reverse embedding).

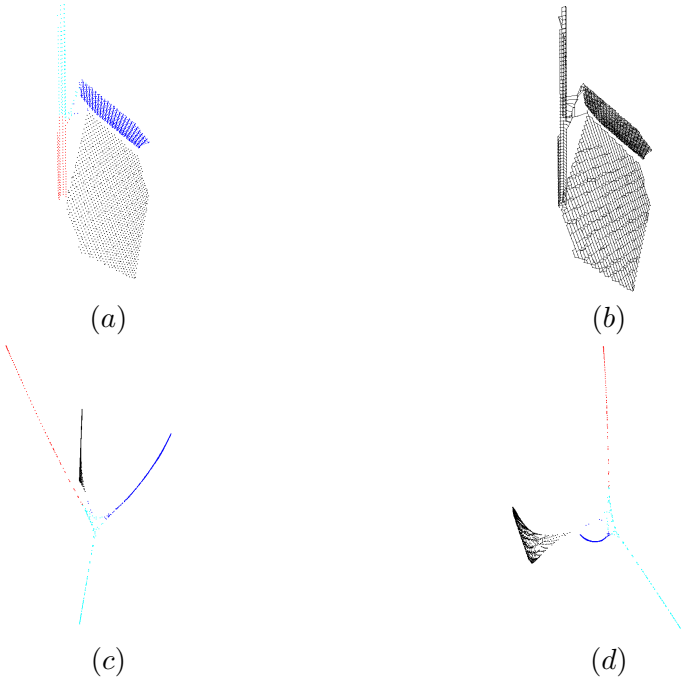



Figure 2. Example of the spectral embedding process. (*a*) Input scan. (*b*) Computed graph. (*c*,*d*) Point cloud embedded into a 3-D spectral space (two different views; see also the accompanying video). Colours are set to roughly indicate the elementary units.

### *2.1.1. Parameters*

The algorithm uses three parameters, one for each stage:

(1) the minimum angle $a$ between two neighbours of any point in the point cloud, for the graph construction;
(2) an estimate $d$ of the number of intrinsic directions in the plant, for the spectral embedding;
(3) the number $c$ of desired subsets of points (elementary units), for the clustering stage.

### *2.1.2. Graph construction*

The first stage of the method aims at recovering neighbouring information between points. This is a difficult task since the only information available is the 3-D location of the points.

Usual methods create neighbouring relationships, called edges, between any point $p$ and either all points which lie within a sphere of radius $\varepsilon$ centred at $p$, or the $k$ nearest points (Figure 3 (a,b)). These methods are known as the $\varepsilon$-Neighbourhood and the $k$-Nearest-Neighbours methods, respectively (Yang (2005); von Luxburg (2007)). $\varepsilon$ and $k$ are user-chosen parameters. $\varepsilon$-Neighbourhood is for example used by Belton et al. (2013), while the $k$-Nearest-Neighbours method is used by Côté et al. (2009); Lou et al. (2015). These methods are convenient so

long as the density of the point cloud is uniform, which is not the case for our TLS data. For non-uniform samples, many redundant edges may be created or relevant ones may be missed and the main problem is to find the right value for the parameters. This problem is shown on Figure 3 $(a)$, where the $\varepsilon$-neighbourhood of a blue point is depicted for two different values of $\varepsilon$ (in green and in red and green, respectively). Similarly, the $k$-nearest-neighbours, for $k = 2$ (in green) and $k = 5$ (in red and green), are shown on Figure 3 $(b)$. If $\varepsilon$ or $k$ is low, the corresponding methods may miss relevant edges, such as the one between the blue point and the upper red point. If $\varepsilon$ or $k$ is high, they may create redundant edges such as the ones between the blue point and the left and right red points.

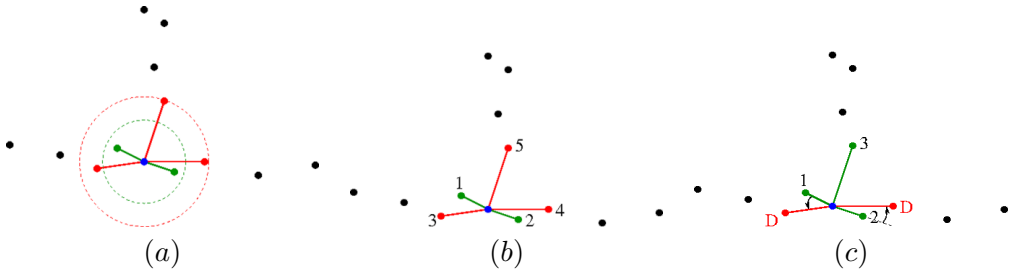



Figure 3. Examples of graph construction for the methods: $(a)$ $\varepsilon$-neighbourhood of a point in blue, for two different values of $\varepsilon$. $(b)$ $k$-nearest-neighbours, for $k = 2$ and $k = 5$. $(c)$ Proposed method, for $a = 45°$.

Note that it is critical for the next stage of our approach, both in terms of memory usage and computation time, to avoid redundant neighbouring relationships since during this stage we work with an adjacency matrix computed from the neighbouring graph. Therefore, the lower the number of neighbours for a given point the sparser the matrix thus the faster the computation. This is why we have developed a specific algorithm which is summarised in Algorithm 1 and Figure 3 $(c)$.

This algorithm starts by selecting a number $k$ of candidate neighbours for every point $p \in P$. In practice, we choose $k = 0.1\%$ of the total number of points. Then, in order to select the neighbours within the set of candidates, it uses one parameter which is the minimum angle $a$ at $p$ allowed between two edges with endpoint $p$ (see Figure 3 $(c)$). If many candidates lie in the same direction, only one (the closest) is selected as a neighbour of $p$. This prevents the creation of redundant, almost parallel, edges. On the contrary, this algorithm having found the closest point in a given direction will go on to look for other points farther away but in a distinctly different direction. Thus, this method captures all relevant edges and is robust to non-uniform density within the point cloud. Figure 3 $(c)$ shows the result of the method for $a = 45°$. This method discards the two red points $D$ since the corresponding edges are within a small angle of existing edges (in green) but it does capture point 3 which is a neighbour in a clearly different direction.

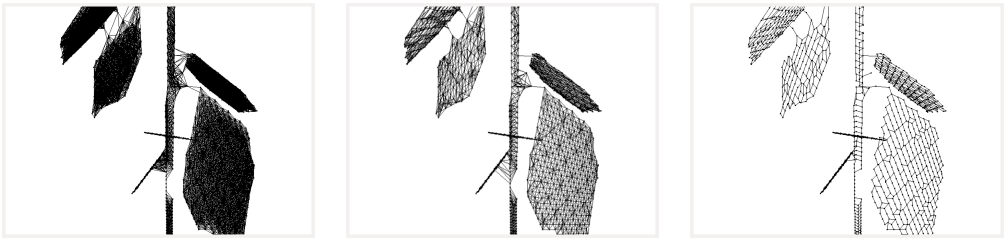
The graph construction runs this algorithm for every point $p$ in the point cloud. The resulting neighbouring graph is thus the union of the selected edges $E(p)$ for every $p \in P$. Note that we do not look for mutual nearest neighbours: if $q$ is computed as a valid neighbour of $p$ but $p$ is not considered as a valid neighbour of $q$, we still connect these two points. Our experimental results have shown that choosing the angle parameter $a = 90°$ is a good compromise in practice (see Section 3.2.3). This allows us to search for neighbours in the 3 cardinal directions around a point in 3D, which has been shown to be sufficient for building

**Data**: Point cloud $P$, a point $p \in P$, a user-chosen angle parameter $a$ (in radian)
**Result**: Set $E(p)$ of the edges of the graph with endpoint $p$
$E(p) := \emptyset$;
Compute the $k$ nearest neighbours of $p$ in $P$, and put them in a priority queue $Q$ ordered by increasing distance to $p$;
**for** $p' \in Q$ **do**
  **if** $\exists e \in E(p)$ *such that* $angle(pp', e) < a$ **then**
    Discard $p'$;
  **end**
  **else**
    Put the edge $pp'$ in $E(p)$;
  **end**
**end**

**Algorithm 1:** Building the neighbouring edges of a single point $p$ in the cloud

a connected graph with as few as possible redundant edges.

Figure 4 shows an example of a graph construction. In this example, the petioles were very sparsely scanned compared to the leaf-blades and the main branch. For the $\varepsilon$-neighbourhood and the $k$-nearest-neighbours methods, the minimum value of the parameter was chosen such that the resulting graph was connected. Notice how both methods, contrary to ours, create numerous redundant edges on the main branch.



($a$) ($b$) ($c$)

Figure 4. Example of graph reconstruction by ($a$) the $\varepsilon$-neighbourhood method (with $\varepsilon = 0.006m$); ($b$) the $k$-nearest-neighbours method (with $k = 8$) and ($c$) the method used in this study ($a = 90°$).

Other methods, which guarantee connectedness of the graph, are proposed by Yang (2005). However, their computational complexity (at least $O(n^2)$, where $n$ is the number of points) may become prohibitive in the context of this study. The approach proposed in Algorithm 1 reaches a $O(n \log n)$ complexity with appropriate data structures, i.e. a kd-tree for the $k$-nearest-neighbours searches and heaps for the priority queues.

### *2.1.3. Spectral embedding*

In the second stage, the major intrinsic directions of the shape are recovered and the weights of the edges modified accordingly. This is done using a technique called dimension reduction or spectral embedding. Indeed, embedding a (discrete) shape into a low-dimensional spectral space is known to help recover its intrinsic features (see e.g. Reuter et al. (2006)). In this work, we build on the Laplacian Eigenmaps framework of Belkin and Niyogi (2003), the main differences being

the graph construction approach described above and the choice of the distance between neighbouring points. This framework is now described.

Let $A$ be the adjacency matrix of the graph constructed in the previous stage. Points are numbered from 1 to $n$, $A$ is a $n \times n$ matrix such that $A(i, j)$ is equal to the weight of the edge connecting points $i$ and $j$. $A(i, j) = 0$ if there is no edge between these points. The Euclidean distance between $i$ and $j$ is used as a weight. Let $W$ be the diagonal valency matrix of the graph. $W(i, i)$ is equal to the sum of the weights of edges with endpoint $i$. The matrix $L = W - A$ is called the Laplacian matrix of the graph. The spectral embedding of the graph into a $d$-dimensional space is given by the $d$ eigenvectors $V_1, \ldots, V_d$ of $L$ associated with the first $d$ non zero eigenvalues (in increasing order). Namely, the embedding coordinates of point number $i$ are given by row $i$ of the matrix whose columns are vectors $V_1, \ldots, V_d$ (von Luxburg (2007)).

It is known that the eigenvectors associated to the lowest non zero eigenvalues of $L$ give the main "intrinsic" (curved) directions of the graph (Lévy (2006)). This property has previously been used for shape compression (Karni and Gotsman (2000)), progressive reconstruction (Lévy (2006)) and deformation (Dey et al. (2012)) purposes. The Laplacian spectral embedding is also known as the eigen skeleton of the input graph (Dey et al. (2012)). Using this property makes sense in the context of this study, since a plant is a strongly anisotropic shape; the natural directions of the plant follow the directions of each stem, branch, petiole and the main directions of each leaf-blade. It is therefore expected that the spectral embedding of the graph aligns points into a curve, or at least a strongly anisotropic shape, that samples each elementary unit of the plant, as shown on Figure 2. It is easier to segment the spectral embedding of the graph into subsets of points than the TLS data, since it does not depend on the particular shape of the leaves. Moreover, geometrical noise accumulated during the acquisition process is implicitly altered by the spectral embedding.

Note that computing the eigen-decomposition depends on the number of edges in the graph. The lower number of neighbours a point has, the sparser the matrix is, thus the faster the computation is. This is why the algorithm described in Section 2.1.2 is used rather than the standard $\varepsilon$-neighbourhood or $k$-nearest-neighbours methods. Figure 2 shows the 3-D embedding of a simple plant model with two leaves, thus having three main directions. Notice how the plant is nearly collapsed to a set of curves.

### *2.1.4. Spectral clustering*

This stage clusters points into sets corresponding to the plant's elementary units (internodes, petioles and leaf-blades). To this aim, the point cloud is segmented according to its spectral embedding; the objective is thus to cluster together points of an elongated curve in the embedded shape. Since the embedded point cloud is almost a set of elongated curves (see Figure 2), segmentation techniques for stems such as e.g. Paulus et al. (2014) could be applied. However, they would not benefit from the point neighbourhood information retrieved from stage one of the approach (Section 2.1.2).

The usual clustering technique, applied in spectral space, is known as K-means clustering (von Luxburg (2007)). It is used for example by Lou et al. (2015).

K-means clustering randomly selects $K$ initial "means" among the points, with $K$ being a user-defined parameter. Each point is assigned to the nearest mean. Then, for each cluster of points, the closest point to the centroid (centre of mass) of the cluster is computed and selected as the new mean. The process is iterated until convergence to stable mean positions is reached, which is generally fast. This technique is well adapted to isotropic data, i.e, a point cloud without any principal direction. This is obviously not the case in this study where the graph is embedded in spectral space almost as a set of elongated curves. More general approaches such as expectation maximisation could be used, but as K-means clustering they do not naturally benefit from the neighbourhood information (graph edges). Note that Lou et al. (2015) merge neighbouring clusters with similar normals, but this may lead to undersegmentation since different elementary units (e.g., two leaves) may have similar normals.

A new clustering method, more adapted to elongated shapes, is therefore proposed. This method is described in Algorithm 2 and Figure 5. The idea is to compute the main directions of the graph (in spectral space), as sets of edge-connected points which are called the *segments*. As many segments as the desired number $c$ of clusters are computed. Finally, each point of the graph is labelled according to its closest segment. Note that $c$ should be odd, by construction. In case the desired number of clusters is even, we recommend to segment in $c+1$ clusters and merge two of them.

**Data**: Graph $G = (V, E)$ (in spectral space), desired number $c$ of clusters
**Result**: Segmentation of $V$ into disjoint sets $\{Cluster[1], \ldots, Cluster[c]\}$
$Source :=$ farthest point to a random point of $G$;
$i := 1$;
$Segment[i] :=$ ComputeShortestPaths($Source$,$G$);
**while** $i < c$ **do**
  $Segment[i+1] :=$ ComputeShortestPaths($Segment[1..i]$,$G$);
  $p :=$ point of $Segment[1..i]$ connected to $Segment[i+1]$;
  $j :=$ number of the segment to which belongs $p$;
  Remove successive points of $Segment[j]$ from $p$ to one of its end and add them to $Segment[i+2]$;
  $i+ = 2$;
**end**
ComputeShortestPaths($Segment[1..c]$,$G$);
**for** $p \in V$ **do**
  $p' :=$ closest point of $Segment[1..c]$ from $p$;
  $j :=$ number of the segment to which belongs $p'$;
  Add $p$ to $Cluster[j]$;
**end**

**Algorithm 2:** Proposed graph segmentation method (applied in spectral space).

### *2.1.5. Algorithmic details*

Edges of the computed graph are weighted by a distance between their two endpoints called the commute-time distance, which represents the expected time for a random walk on the graph to travel from one point to the other and then return (Qiu and Hancock (2007)). In our plant segmentation context, this is a

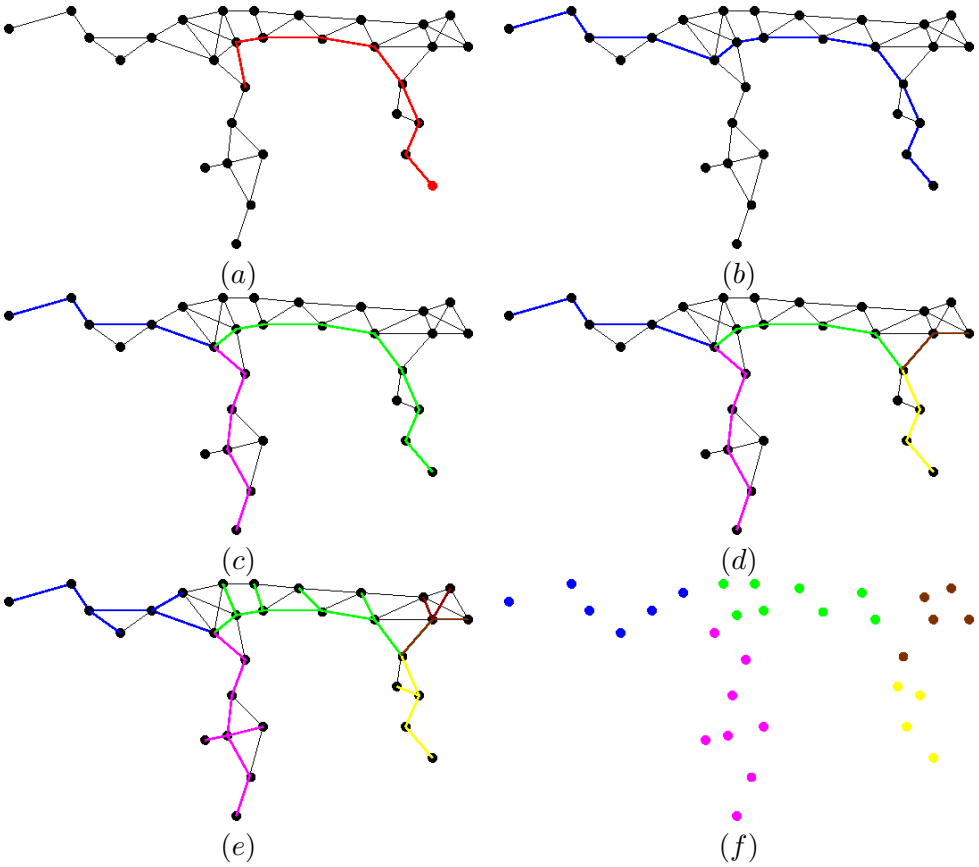



Figure 5. The segmentation process. (*a*) Input graph (in spectral space) and selected source point (in red, together with the path from the initial random point). (*b*,*c*,*d*) Computation of successive segments. (*e*) Shortest paths from each remaining point to the segments. (*f*) Computed clusters.

more meaningful distance than the Euclidean distance. For example, points on two different leaf-blades connected by a few edges (see Figure 6) may have a short Euclidean distance and a large commute-time distance in the graph. Since we want such points to belong to different clusters, we want their distance to be large. Moreover, commute-time distance has been proved to be robust against noise for clustering purposes (Qiu and Hancock (2007), Sec. 5.1).

The commute-time distance is similar to the Euclidean distance in spectral space, except each coordinate is divided by the corresponding eigenvalue. More precisely, the commute-time distance between points $i$ and $j$ is given by $\sqrt{\sum_k \frac{(i(k)-j(k))^2}{e(k)}}$, with $i(k)$ and $j(k)$ the $k$-th coordinates in spectral space of $i$ and $j$, respectively (that is to say, the $i$-th and $j$-th coordinates of the $k$-th eigenvector $V_k$ of the Laplacian matrix of the graph, as explained above), and $e(k)$ the $k$-th eigenvalue of the Laplacian matrix of the graph (Qiu and Hancock (2007), Sec. 2.3).

Finding the main segment of a weighted graph is a typical issue in medial structure axis and skeleton-related problems. The main segment of a graph can be computed successfully by using a one-source shortest path algorithm from an endpoint of the graph, e.g. the Dijkstra's algorithm (Dijkstra (1959)). This endpoint can be found as the farthest point to some random point (Lazarus and Verroust (1999)) and computed once again using a one-source shortest path algorithm. Other seg-

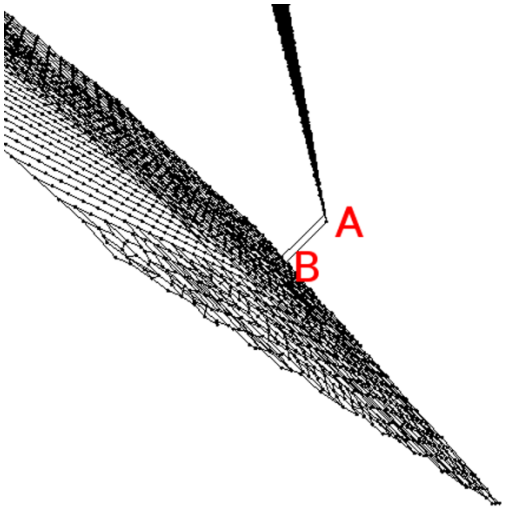



Figure 6. Points $A$ and $B$ are on two different leaf-blades of the poplar mock-up with high TLS resolution level (see Section 2.2.1). Their commute-time distance in the graph is large while their Euclidean distance is small.

ments are then computed the same way by taking all points of already computed segments as source points as in Hassan et al. (2011). As a result, each point of the graph is linked to its closest point on the segments and its distance to this point is computed. See Figure 5 for an example.

#### *2.1.6. Asymptotic computational complexity*

As explained above, Algorithm 2 uses a one-source shortest path algorithm $(c-1)/2+3$ times. Then, a Disjoint Set data structure (Cormen et al. (2009)) is used to cluster and label the points according to their closest point on the segments. The computational complexity of Dijkstra's algorithm, using a heap data structure, is $O(m+n\log n)$, where $n$ is the number of points in the graph and $m$ is the number of edges. The complexity of cluster creation within a Disjoint Set framework and using relevant heuristics is $O(n\log n)$ (Cormen et al. (2009)). The computational complexity of Algorithm 2 is thus $O(c(m+n\log n))$.

### 2.2. *Reference data*

Reference point clouds were obtained at various resolution levels, from two different ways. First, point clouds were generated from a virtual poplar seedling mock-up through a computer simulation of TLS (Section 2.2.1). Second, points clouds were acquired from four real plants using a Leica Geosystems HDS-6100 TLS device (Section 2.2.2).

#### *2.2.1. Point cloud computations from a poplar mock-up*

The 3D structure of a one-year-old single-stem seedling of poplar clone Trichobel (*Populus trichocarpa* Torr. & Gray x *P. trichocarpa*) was generated by the 3D Coppice Poplar Canopy Architecture model (3D CPCA) developed by Casella and Sinoquet (2003) (Figure 7, Table 1). The model is based on a multi-scale decomposition of a plant structure into components (axis and growth unit) described as a collection of metamers, themselves defined as a collection of elementary units (nodes plus internodes, petioles and leaf-blades) (Godin et al.

(1999)). For this study, axes (stem and branches) were divided as a sequence of conical frustums (a sequence of internode units), petioles were represented as cylinders and leaf-blades were regarded as planar objects. Each elementary unit was scaled to the appropriate geometric dimensions (e.g. height, base and top radius for a conical frustum) although a leaf-blade prototype was created and represented as a polygon with a set of 4 contiguous triangles to fit the leaf-blade shape and the allometric relationships between the leaf-blade area, the leaf-midrib length and the leaf-blade width. Each unit was them rotated and translated according to its orientation and location in the scene. Each unit was scaled so that no discontinuity between elements was possible, and there were no contact between laminae. Empirical functions and random deviation used in this study for the reconstruction of the plant architecture were as in Casella and Sinoquet (2003). The resulting poplar mock-up consisted of 17 leaves, 17 petioles and 24 internodes.

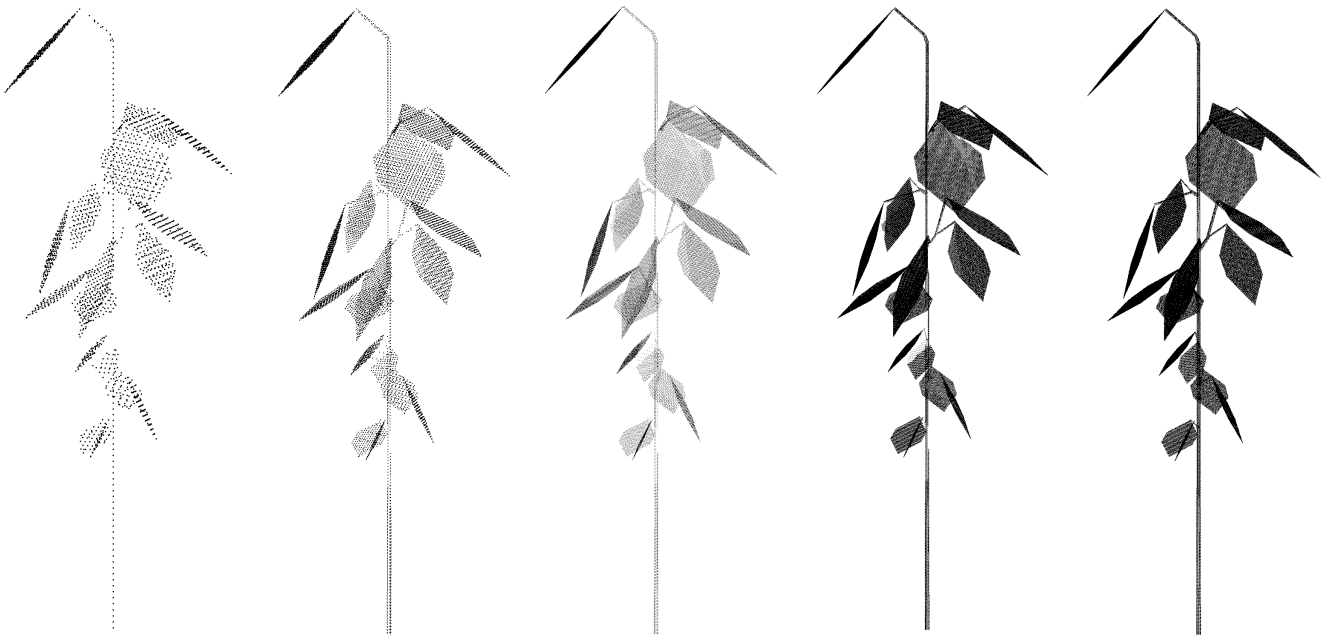



Figure 7. 3-D point cloud images of the poplar seedling mock-up used in this study for the low (L), medium (M), high (H) and ultra high (U) TLS resolution levels (Table 2). The last image shows the point cloud generated without simulation of the occlusion: all hits from the laser source to objects were recorded.

| Seedling | Height ($m$) | Nb. of leaves | Total leaf area ($10^{-2}m^2$) |
|---|---|---|---|
| Poplar mock-up | 0.462 | 17 | 2.617 |
| Birch | 0.650 | 7 | 1.296 |
| Horse chestnut | 0.607 | 9 | 16.567 |
| Sweet chestnut | 0.465 | 19 | 4.530 |
| Red oak | 0.547 | 10 | 5.553 |

Table 1. Structure parameters of the tree seedlings.

This mock-up could then be scanned from any point of view, after having placed a virtual TLS in the scene. Point clouds were computed for three TLS positions around the mock-up and for four scanner resolution levels i.e. by simulating the characteristics and settings of a Leica Geosystems HDS-6100 TLS device (Table 2)

used in this study for point cloud acquisitions from real plants (see next Section). The positions of the virtual TLS in the scene were computed for a distance of 3 meters from the base of the stem to the laser source, an elevation angle of 25° and an azimuth angle of 0, 120 or 240°. For each TLS resolution level, a point cloud was generated using a simple hit/not hit determination algorithm coded from a set of ray/objects (i.e. /cylinder, /cone and /triangle) intersection equations (see Haines (1989)) by determining either the closest or all hits from the laser source along any simulated ray trajectory within the scene. Every recorded hit $(x_{hit}, y_{hit}, z_{hit})$ was then referred to both its related object in the scene (e.g. leaf-blade #) and the position of the laser source $(x_{TLS}, y_{TLS}, z_{TLS})$. Point clouds were generated both with and without simulation of the occlusion, i.e. recording either only the closest or all hits from the laser source to objects.

| Single shot phase-shift technology with single return signal | | | | |
|---|---|---|---|---|
| Wavelength ($nm$) | $650 - 690$ | | | |
| Range ($m$) | $0.3 - 50$ at 18% albedo | | | |
| Spot size at exit ($m$) | 0.003 | | | |
| Beam divergence (°) | 0.0126 | | | |
| Pre-set scanner resolution levels | Low (L) | Medium (M) | High (H) | Ultra high (U) |
| Angular sampling resolution (°) | 0.072 | 0.036 | 0.018 | 0.009 |
| Maximum point spacing at $3m$ ($m$) | 0.0038 | 0.0019 | 0.0009 | 0.0005 |

Table 2. Characteristics and settings of the Leica HDS-6100 terrestrial laser scanner used in this study for point cloud acquisitions.

### *2.2.2. Point cloud acquisitions from real tree seedlings*

TLS point clouds were acquired from tree seedlings of silver birch (*Betula pendula* Roth), horse chestnut (*Aesculus hippocastanum* L.), sweet chestnut (*Castanea sativa* Mill.) and red oak (*Quercus rubra* L.) (Table 1) using a Leica Geosystems HDS-6100 TLS device (Table 2). Each seedling was scanned inside a large area warehouse from three TLS positions around the plant (two for the Horse chestnut seedling) and for three scanner resolution levels. These species were chosen in order to get seedlings with varying structural and leaf geometrical complexities.

TLS point clouds of trees are usually noisy because of various interferences during the acquisition process, see e.g. Hebert and Krotkov (1992). Each point cloud was thus filtered in order to remove most of the outliers, using the statistical outlier removal filter of the Point Cloud Library (Rusu and Cousins (2011)). For each point $p$, its $k$ nearest neighbours were first retrieved, and the mean distance $d$ of these points to $p$ was computed. If $d$ is outside an interval defined by the mean and the standard deviation of the mean distances to all points, then $p$ is detected as an outlier and removed. We have set $k$ so that to remove as many outliers as possible, without removing relevant points. $k = 20$ has been taken for the horse chestnut, $k = 30$ for the sweet chestnut, and $k = 50$ for the silver birch and the red oak, for all resolution levels. Between 0.2% (red oak) and 11% (silver birch) of the points have been removed by this filtering. Table 3 shows the number of point in each point cloud after filtering. Each resolution level contains about 4 times more points than the previous one.

| Seedling | Low (L) | Medium (M) | High (H) | Ultra high (U) |
|---|---|---|---|---|
| Poplar mock-up | 2452 | 10244 | 40560 | 162704 |
| Birch | 3230 | 14412 | 60517 | - |
| Horse chestnut | 6691 | 12615 | 122022 | - |
| Sweet chestnut | 9761 | 38111 | 155186 | - |
| Red oak | 12667 | 50997 | 187054 | - |

Table 3. Number of points for each point cloud (after filtering).

### 2.3. *Statistical analysis of leaf area estimates on the poplar mock-up*

Leaf area (LA) has been estimated for each labelled leaf-blade, by projecting its points into the least-square fitting plane, computing the Delaunay triangulation of the projected points (Edelsbrunner (2001)), projecting the points back to their original positions and summing the areas of the Delaunay triangles. This was done for the leaf clusters as labelled in the input data, both without and with occlusions, as well as for the clusters computed with the presented algorithm. The quality of the method was then determined by two parameters, the root mean square error ($RMSE$) and bias ($b$), defined as:

$$RMSE = \sqrt{\frac{\sum_{k=1}^{n} (\hat{y}_{pk} - y_{ak})^2}{n}}$$

$$b = \sum_{k=1}^{n} \frac{(y_{sk} - y_{ak})}{n}$$

where $n$ is the number of observations and $\hat{y}_{pk}$ is the predicted average value from the regression line between the simulated $y_{sk}$ and the actual $y_{ak}$ values for the $k^{th}$ observation.

## 3. Results and discussion

The algorithm has been implemented in C++ and Matlab. Because of the generally complex structure of a plant, perfect clusters may not be created in a single run. In practice, the algorithm is first run with a low number of desired clusters (less than the actual number of metamers), then each cluster is segmented by running this algorithm again. A simple graphical user interface has also been implemented, which allows merging clusters by selecting a point in each cluster. The overall approach is thus semi-automatic.

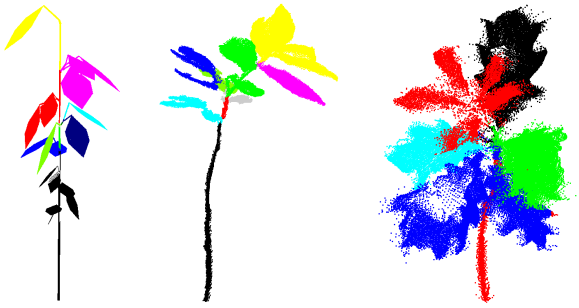
Results of the segmentation process on the poplar mock-up and on the four tree seedlings are shown in Section 3.1. A quantitative validation is provided in Section 3.2. It includes an evaluation of the segmentation accuracy, a statistical analysis at the leaf area scale, and a parameter sensitivity analysis of the algorithm.

### 3.1. *Qualitative results*

The method has been tested on the five different tree seedlings (Tables 1 and 3), for the low, medium and high resolution levels (Table 2), as well as the ultra high resolution level for the poplar mock-up.

#### *3.1.1. First segmentation*

Results of the first run of the algorithm are shown on Figure 8 for the poplar mock-up (ultra high resolution), the sweet chestnut (high resolution) and the red oak (medium resolution) seedlings. When a small number $c$ of clusters is set, the algorithm usually segments the point cloud into connected subsets of elementary units, even when the point cloud is very noisy (e.g., the red oak). The higher value for $c$, the higher probability that a elementary unit (usually, a leaf) is segmented by the algorithm into several clusters (see Figure 9). We elaborate on the choice of $c$ in Section 3.2.3.



($a$) ($b$) ($c$)

Figure 8. Segmentation results after the first run for ($a$) the poplar mock-up with occlusions, ($b$) the sweet chestnut and ($c$) the red oak, with $c = 11$ ($a$), $c = 9$ ($b$) and $c = 5$ ($c$), respectively.

Once the initial point cloud has been segmented, the user can select any given cluster through the graphical interface and re-run the algorithm on this cluster. This is done interactively; no botanical knowledge is used in our approach and the user decides which subsets of points to segment and when to stop this process.

#### *3.1.2. Final segmentations*

Qualitative final segmentation results on all scans are shown on Figures 10, 11, 12 and 13. The accompanying videos also show the segmentation results for the high resolution level point cloud of each of these five seedlings.

These results show that overall, despite large occlusions in real scans (see e.g. Figures 12 ($d$) and 13 ($b$)), the method correctly segments the point cloud into sets of individual leaf-blades, petioles and stem sections. Internodes can be detected when both ends are delimited by petioles and/or incident stems, otherwise they are merged. The method is insensitive to the leaf anatomy. It behaves correctly for both simple, small (e.g. sweet chestnut) and complex, large (e.g. red oak) leaves, as well as for both planar and curved leaves. However, a compound leaf is segmented into its leaflets, as shown for the horse chestnut, as each leaflet corresponds to a different intrinsic direction.

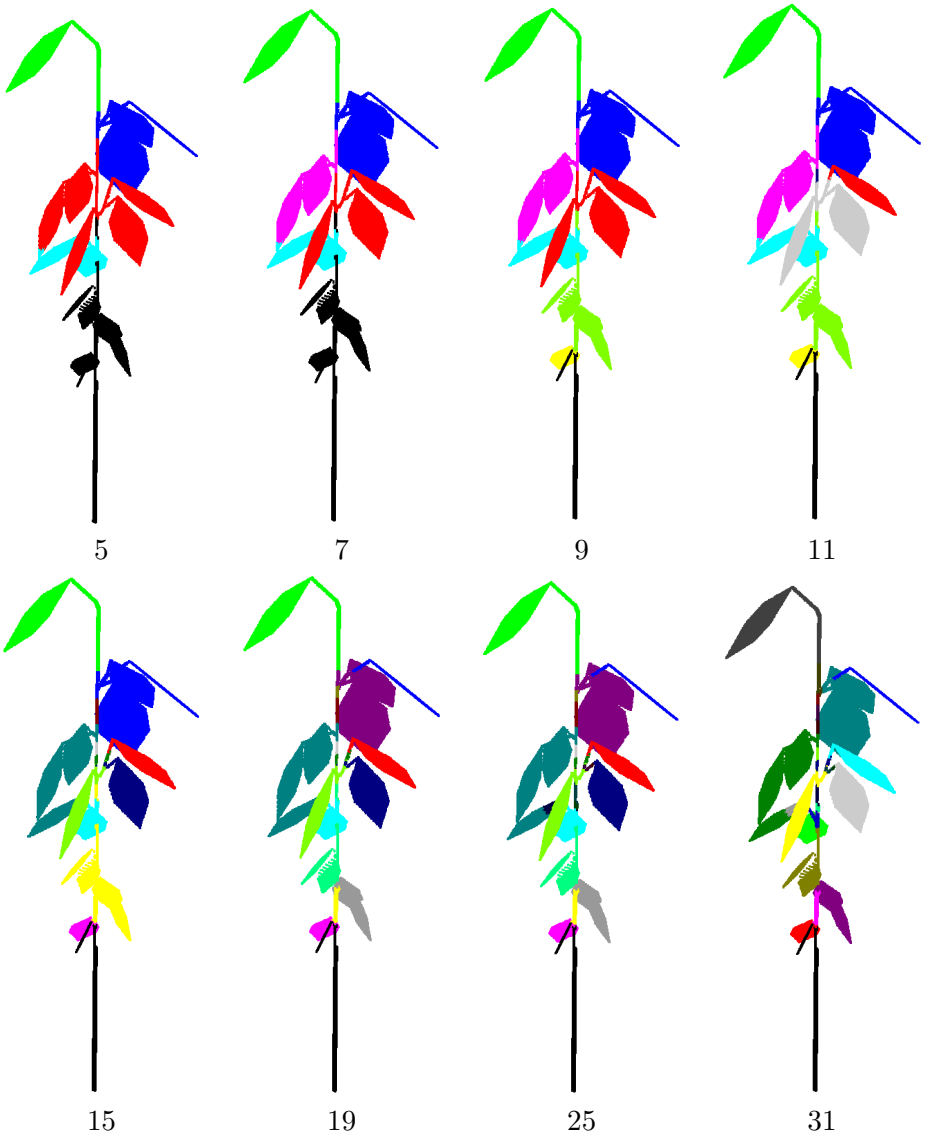



Figure 9. Result of the first iteration of the algorithm for various numbers $c$ of clusters.

Figures 10 and 11 show that the resolution level does not have a strong influence on the segmentation, as will be demonstrated in Section 3.2. The algorithm is also robust to non uniform density within a point cloud, as shown for example on Figure 12 ($a$). Finally, the method is insensitive to the noise level. Even when points are spread over the boundaries of a unit (leaf or stem), they are included into the correct cluster (see Figures 12 ($a$) and 13 ($c$)). This is also shown by the following experiment.

### *3.1.3. Robustness to acquisition noise*

In order to test the robustness of the approach, a raw scan of the sweet chestnut (high resolution level) from a single viewpoint has also been segmented. 72754 points belong to this unfiltered point cloud. Results are shown on Figure 14, to be compared with Figures 11 (H) and 12 ($c$). Points are correctly assigned to their corresponding cluster, except on ambiguous areas (for example between two neighbouring leaves).


Figure 10. Segmentation results for the poplar mock-up with occlusions. The letter indicates the resolution level. On each point cloud, all points with the same colour belong to the same cluster.

## 3.2. *Evaluation*

### *3.2.1. Segmentation accuracy*

The number of points correctly assigned to each elementary unit has been retrieved on the poplar mock-up. We call *false positive* (FP) for a given cluster a point that is labelled as part of this cluster by the segmentation algorithm, while it does not belong to this cluster in the input poplar mock-up point cloud. A point is a cluster *false negative* (FN) if it is not labelled as part of this cluster, while it actually belongs to it. A cluster's false positive rate (FPR) is the ratio of false positives over the actual number of points in the cluster. We define false negative rates (FNR) the same way.

It is worth mentioning that all points were labelled by the algorithm. This is because the constructed graph contains all points of the point cloud, and the spectral segmentation algorithm browses the whole graph. Figure 15 (*a*) shows that a large majority of leaf points are assigned to the correct leaf cluster, the worst case being cluster 5 (the bottom red leaf on Figure 10) in the ultra high resolution level point cloud with 5.77% of points assigned to another cluster. As shown on Figure 15 (*b*), leaf false positive rates are similar to false negative rates. The maximum is reached for cluster 2 in the ultra high resolution point cloud, which correspond to the bottom dark blue leaf on Figure 10, with 5.84% of false positives. False positives and negatives usually occur near the junction of a leaf to its petiole. The segmentation is not always accurate for the petioles and the internodes. This is explained by the fact that several sets of internodes and/or petioles are not fully segmented into elementary units, thus points of different internodes are assigned to the same cluster. This is for example the case of clusters 35 to 39, which correspond to internodes of the poplar's basis stem. Since no geometrical feature enables to split the stem into its internodes, and since the algorithm does not use any botanical knowledge, points are not segmented into internode clusters and remain in one global cluster, in green on Figure 10.


Figure 11. Segmentation results for point clouds of four contrasting plant seedlings with three different TLS resolution levels each. From top to bottom: birch, horse chestnut, sweet chestnut and red oak. On each point cloud, all points with the same colour belong to the same cluster.

Results are summarised in Table 4, in which we have computed means and standard deviations of the number of points over leaf, petiole and internode clusters, respectively. It shows that for leaves, false positive and negative rates remain below 3.4%. However, since it is difficult to unravel some petioles or internodes to their adjacent units from a pure geometrical point of view, points of neighbouring petiole or internode clusters are often pooled together. As a result, many petiole or internode clusters have no point assigned, leading to huge false positive and negative rates. Table 4 also shows that the resolution level has little impact on the segmentation accuracy, although results are slightly better for low resolution point clouds than for high resolution ones.

### *3.2.2. Statistical analysis of the leaf area estimates*

Results of the leaf area estimates for the poplar mock-up are shown in Table 5. They show that the resolution level has a stronger influence on leaf area estimates than our segmentation method. Our estimates are always close to the estimates computed for the correct clusters.

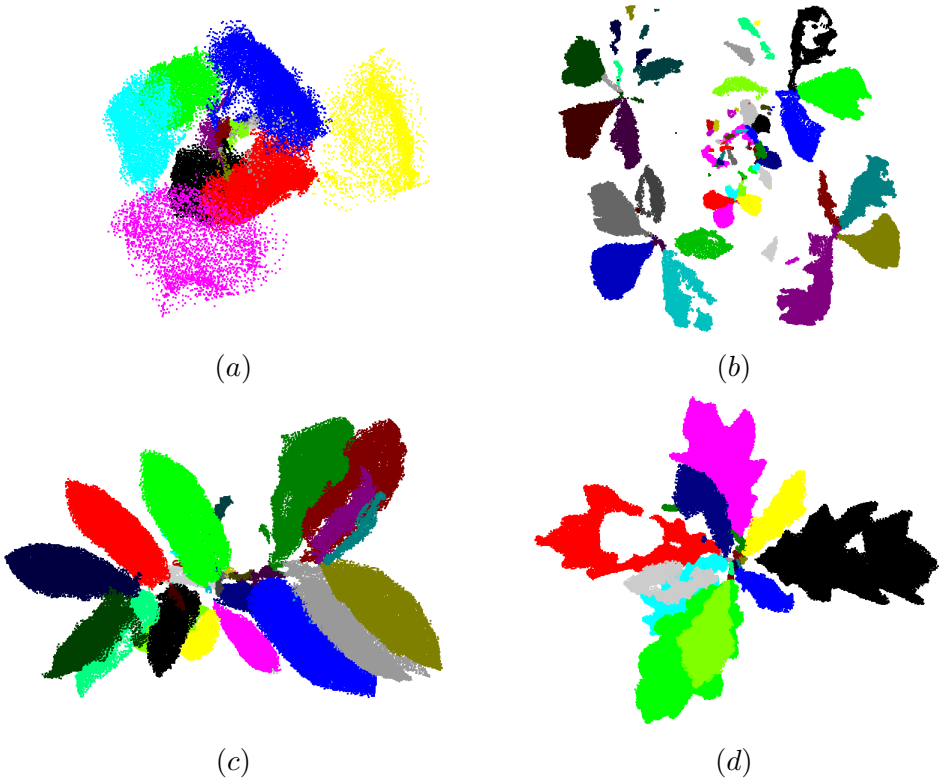


Figure 12. Segmentation results (top view). (*a*) Birch. (*b*) Horse chestnut. (*c*) Sweet chestnut. (*d*) Red oak. All are high TLS resolution level point clouds.

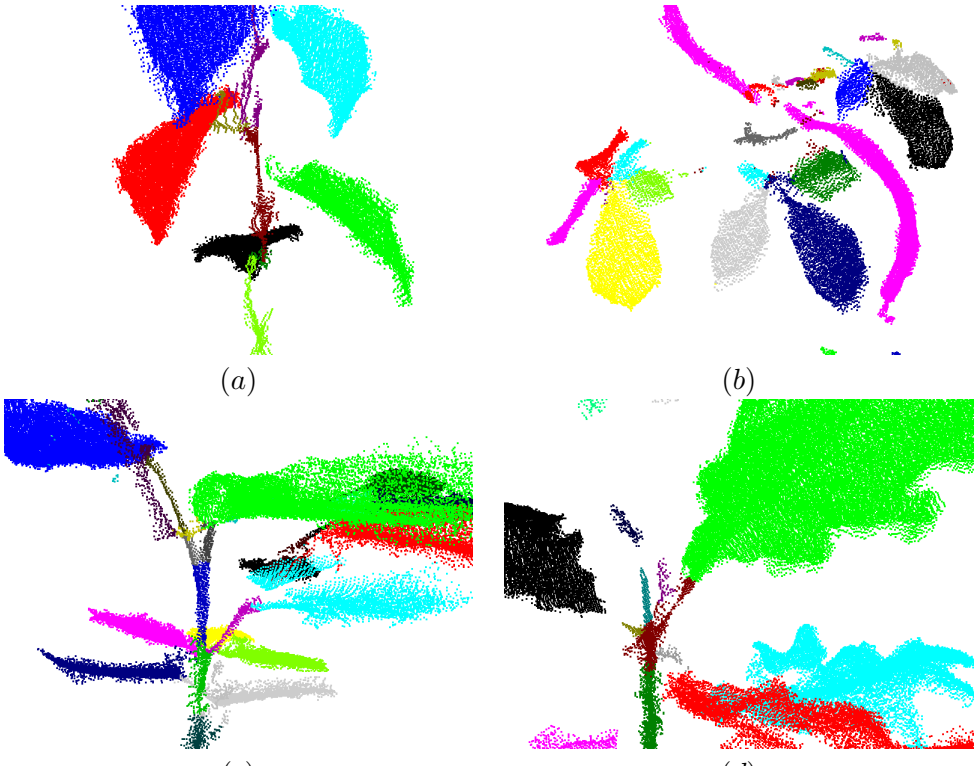
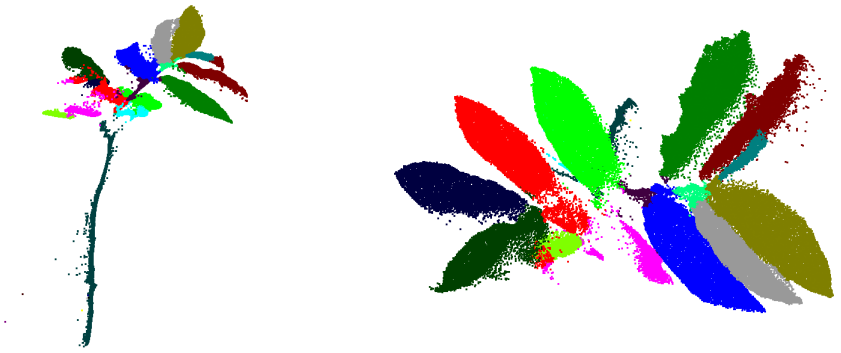
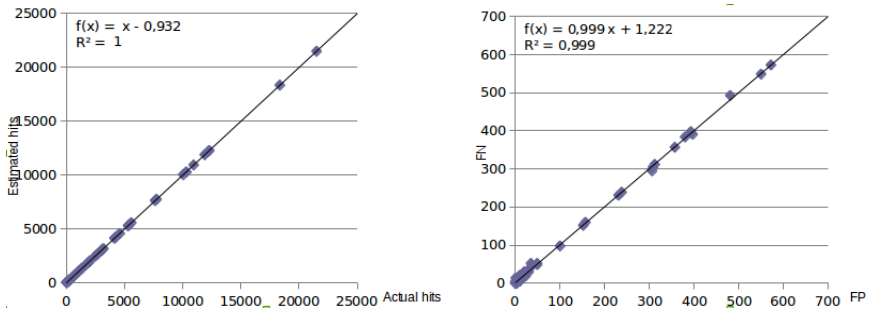


Figure 13. Segmentation results (close ups). (*a*) Birch. (*b*) Horse chestnut. (*c*) Sweet chestnut. (*d*) Red oak. All are high TLS resolution level point clouds.



(a) (b)

Figure 14. Segmentation results on a raw, unfiltered scan from a single viewpoint of the sweet chestnut seedling. (*a*) Front view. (*b*) Top view. High resolution level point cloud.




(a) (b)

Figure 15. Correlations between (*a*) the number of estimated points vs. the number of actual points and (*b*) the number of false negatives vs. the number of false positives, for all leaf clusters at all resolution levels.

| Resolution | Nb. of points | Leaves | | Petioles | | Internodes | |
|---|---|---|---|---|---|---|---|
| | | Mean | SD | Mean | SD | Mean | SD |
| Low (L) | Actual | 136 | 85 | 3 | 3 | 5 | 4 |
| | Est. | 136 | 84 | 3 | 4 | 5 | 10 |
| | FP | 1 | 1 | 1 | 3 | 3 | 7 |
| | FN | 1 | 1 | 1 | 1 | 3 | 3 |
| Medium (M) | Actual | 548 | 346 | 13 | 12 | 39 | 26 |
| | Est. | 547 | 347 | 14 | 12 | 40 | 83 |
| | FP | 3 | 2 | 3 | 4 | 24 | 62 |
| | FN | 4 | 4 | 2 | 2 | 22 | 25 |
| High (H) | Actual | 2193 | 1382 | 50 | 49 | 133 | 81 |
| | Est. | 2190 | 1380 | 57 | 53 | 137 | 223 |
| | FP | 21 | 11 | 11 | 16 | 74 | 174 |
| | FN | 24 | 13 | 4 | 5 | 70 | 97 |
| Ultra high (U) | Actual | 8773 | 5528 | 196 | 192 | 562 | 338 |
| | Est. | 8773 | 5527 | 200 | 195 | 571 | 1086 |
| | FP | 295 | 163 | 13 | 14 | 285 | 805 |
| | FN | 295 | 163 | 10 | 13 | 276 | 381 |

Table 4. Number of points and false positives (FP) and negatives (FN) for the poplar mock-up. SD stands for *standard deviation*.

| Resolution | Occlusions | Segmentation | Slope | Intercept ($10^{-4}m^2$) | RMSE ($10^{-4}m^2$) | Bias ($10^{-4}m^2$) | Mean LA (± SD) ($10^{-4}m^2$) |
|---|---|---|---|---|---|---|---|
| Low (L) | No | Actual | 0.99 | −1.17 | 0.28 | −1.34 | 14.06 (7.7) |
| | Yes | Actual | 1 | −1.63 | 0.38 | −1.64 | 13.75 (7.7) |
| | Yes | Computed | 1 | −1.19 | 0.90 | −1.25 | 14.15 (7.8) |
| Medium (M) | No | Actual | 1 | −0.33 | 0.10 | −0.39 | 15.01 (7.7) |
| | Yes | Actual | 1 | −0.53 | 0.14 | −0.52 | 14.87 (7.7) |
| | Yes | Computed | 1.01 | −0.64 | 0.37 | −0.43 | 14.96 (7.9) |
| High (H) | No | Actual | 1 | −0.09 | 0.02 | −0.13 | 15.27 (7.7) |
| | Yes | Actual | 1 | −0.21 | 0.08 | −0.18 | 15.21 (7.8) |
| | Yes | Computed | 1 | −0.20 | 0.15 | −0.17 | 15.22 (7.8) |
| Ultra high (U) | No | Actual | 1 | −0.04 | 0.01 | −0.04 | 15.35 (7.7) |
| | Yes | Actual | 1 | −0.09 | 0.04 | −0.06 | 15.33 (7.8) |
| | Yes | Computed | 1.03 | −0.37 | 0.55 | +0.12 | 15.51 (8.0) |

Table 5. Statistical analysis of leaf area estimates on the poplar mock-up, for all resolution levels. Mean LA (± SD) control = 15.39 (7.7) $10^{-4}m^2$. SD stands for *standard deviation*.

Sampling underestimates leaf areas because our area estimation method creates a piecewise linear surface which boundary is defined by points labelled as belonging of the leaf. Since these points are actually fully inside the leaf and not on its boundary, and since leaves of the poplar mock-up are approximated by convex flat surfaces, the computed surface is smaller than the actual one. The higher resolution, the smaller underestimate, since the boundary points for the Delaunay triangulation are closer to the actual leaf boundary. In case of occlusions some points may be missing in a leaf cluster, leading to a smaller surface estimate, thus again an underestimate of the leaf area.

It can also be noticed on Table 5 that our approach tends to slightly overestimate leaf areas with respect to the estimate of the actual segmentation. This is mainly due to the fact that a false positive point may easily add a large area to the estimate, since the Delaunay triangulation will create big triangles between this point and other points in the cluster. This is a counterbalancing effect to the underestimates of the sampling and the resolution.

### *3.2.3. Sensitivity analysis*

We now detail some experiments on the sensitivity of the method to the three parameters. The algorithm has been run on the poplar mock-up with different values for all three parameters, see Tables 6, 7 and 8. We have computed the false positive and false negative rates for each set of parameters, as well as the variation of the estimated total leaf area (−1% means that the estimated total leaf area is 1% lower than the actual leaf area, which is $0.02617m^2$).

| $a$ (°) | 30 | 45 | 60 | 75 | 90 |
|---|---|---|---|---|---|
| Edges | 204195 | 136617 | 100446 | 81855 | 69618 |
| Computation time ($s$) | 376 | 371 | 361 | 367 | 361 |
| Leaf FPR | 0.91% | 0.94% | 0.89% | 0.90% | 0.94% |
| Leaf FNR | 0.98% | 0.96% | 0.92% | 0.97% | 1.10% |
| Signed TLA error | −0.57% | −0.65% | 1.03% | −0.84% | −1.15% |

Table 6. Influence of parameter $a$ on the poplar mock-up (H), with $d = 10$ and $c = 11$. TLA stands for *total leaf area*.

According to the experiments made (Table 6), the total leaf false positive (FPR) and negative (FNR) rates and signed leaf area errors only vary by 0.05%, 0.12%

| $d$ | 5 | 10 | 15 | 30 |
|---|---|---|---|---|
| Computation time ($s$) | 272 | 361 | 439 | 711 |
| Leaf FPR | 0.93% | 0.94% | 0.87% | 0.94% |
| Leaf FNR | 0.95% | 1.10% | 0.95% | 0.93% |
| Signed TLA error | −0.23% | −1.15% | −1.22% | −0.19% |

Table 7. Influence of parameter $d$ for the poplar mock-up (H), with $a = 90°$ and $c = 11$.

| $c$ | 5 | 7 | 9 | 11 | 15 | 19 | 25 | 31 |
|---|---|---|---|---|---|---|---|---|
| Nb. of overseg. leaves | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 |
| Computation time ($s$) | 329 | 340 | 351 | 361 | 384 | 409 | 441 | 478 |

Table 8. Influence of parameter $c$ for the poplar mock-up (H), with $a = 90°$ and $d = 10$.

and 2.18%, respectively, with respect to the angle $a$. As explained in Section 2.1.6, computation time should be affected by the number of edges in the graph, which in turn depends on the value chosen for the angle parameter $a$. However, as shown in Table 6, although the number of edges exponentially decreases with the angle (see also Figure 16), the total computation time does not vary much with $a$. In particular, the computation time for stage 3 is always $91s$. This contradicts the theoretical computational complexity analysis (Section 2.1.6). We explain this counter-intuitive result by the fact that in practice, since our data is a set of elongated shapes in the embedding space, Dijkstra's algorithm does not update the shortest paths much and many edges of the graph are not used. Its complexity in practice is thus close to $O(n \log n)$ rather than $O(m + n \log n)$.

As a conclusion, the method is rather insensitive to parameter $a$. However, in the case the graph is to be stored in a file, we advise to choose a value of $a = 90°$ to reduce its size (see Figure 16). According to our experiments (not shown here), a value of $a$ greater than 90° may lead to a disconnected graph.

The total computation time linearly increases with respect to the number $d$ of intrinsic directions (Table 7). This parameter does not affect much the total false positive and negative rates, which only varies by 0.07% and 0.17%, respectively, and the total leaf area error, which only varies by 1.03%. Therefore, it is not necessary to set a high number of intrinsic directions. Our experiments indicate that $d = 5$ or $d = 10$ are good guesses in most of the cases.

Our experiments (Table 8, Figure 9) show that choosing a large number $c$ of clusters may lead to over-segmentations of leaves. On the poplar mock-up, the bottom leaf is segmented in two different clusters from $c = 9$ (not visible on Figure 9 since this leaf is side-view), and this is also the case for a second leaf from $c = 25$. To overcome this problem, we suggest to first set a small value for $c$. According to our experiments, $c \sim 25\%$ of the total final number of clusters is generally a good guess. If some elementary units are nonetheless over-segmented, we provide a graphical interface to easily select and merge the corresponding clusters. For the examples shown on Figure 8, $c$ was set to $23\%, 27\%$ and $26\%$ of the final number of clusters, respectively ($c = 11, 9$ and 5 for $48, 33$ and 19 final clusters).

Computation time linearly increases with respect to the number $c$ of desired clusters, as shown in Table 8. This is consistent with the previously explained computational complexity analysis (Section 2.1.6). Note that indicated computation times are for the first iteration only.

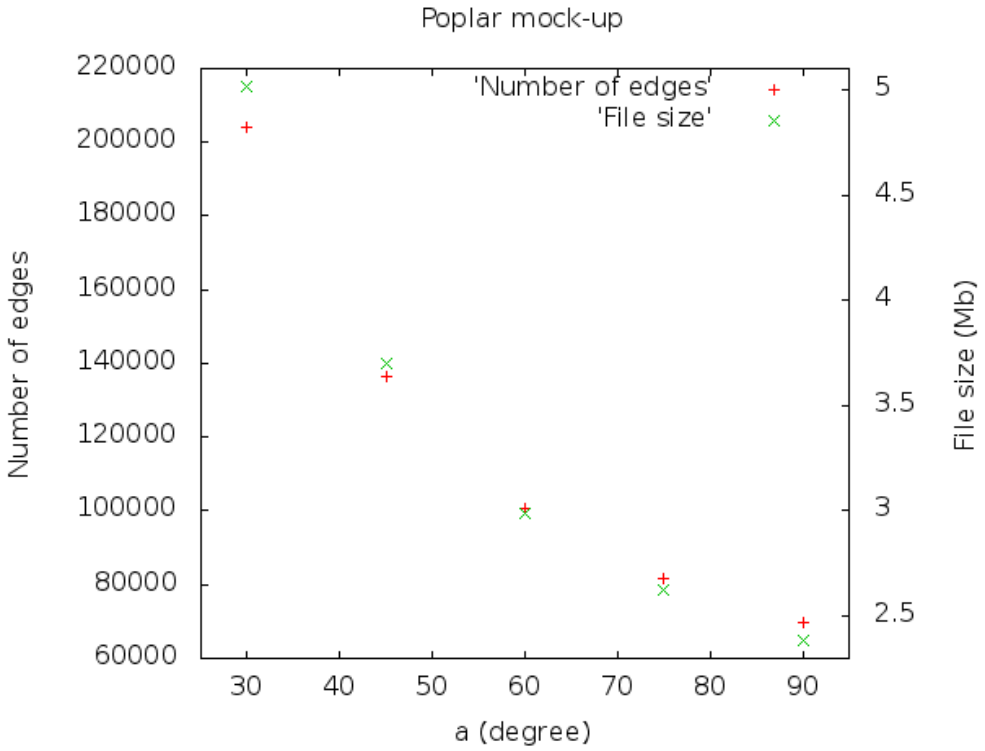



Figure 16. Number of edges in the graph and file size with respect to the chosen angle $a$.

## 4. Discussion and conclusion

We presented here a semi-automatic method to segment a TLS point cloud of a small plant into its elementary units (internodes, petioles and leaf-blades). Qualitative results on four real tree seedlings show that such small scale plants are successfully split into leaf, petiole and stem components. The only two exceptions are compound leaves which are segmented in their leaflets, and adjacent internodes on a stem which may not be separated if no geometrical feature (bud, incident stem) is available. The method does not use any prior botanical knowledge, therefore can be applied in a wide variety of cases. Quantitative results on leaves show that the method is robust (around 1% labelling error) and leads to accurate leaf area estimates.

Only three parameters are used by the method. Only one of them, namely the desired number $c$ of clusters, has an actual influence on the results. According to our tests on four different seedlings, results are insensitive to the branching structure and the leaf anatomy. Acquisition noise during the scanning process is also robustly handled, as shown on the red oak scans. The scan resolution also has little effect on the segmentation result, but has a strong influence on the leaf area computation.

Since no botanical knowledge is used by the algorithm, computed elementary units are not explicitly labelled as leaves, petioles or leaf-blades. This could be done in an additional step with a Principal Component Analysis as in Belton

et al. (2013) or feature based histograms as in Paulus et al. (2013).

The proposed method being semi-automatic, it is suited for small plants such as tree seedlings but may be time consuming for more complex structures. In order to enhance the quality of the segmentation with a large number $c$ of clusters, thus to reduce the interaction time for large scale trees, two improvements are planned in the future. First, we are currently working on the correction of the acquisition noise during the scanning process, in order to reduce the number of points sparsely sampled between elementary units. Filtering the input scans in a pre-processing step, which has been done in this paper, is not a perfect solution since it removes points and thus leads to underestimates of the leaf areas. Second, we plan to enhance the graph construction process (first step of the algorithm), in order to decrease the number of edges between two non adjacent elementary units from a botanical point of view (e.g., two leaves, as in Figure 6). Then, the algorithm will be tested on more complex structures such as full-scale trees.

## Acknowledgements

The authors would like to express their sincere gratitude to the Forestry Commission, the University of Grenoble Alpes and Inria for funding this work, as well as to Rémy Cumont for his participation in coding the segmentation algorithm and Dr Elisa Hétroy-Wheeler for proof-reading the paper.

Author contributions: E.C. and F.H.W. designed the research; F.H.W. designed and coded the segmentation algorithm; E.C. performed the real data acquisition and coded the hit-no hit algorithm for point cloud simulations; D.B. filtered the point clouds; F.H.W. and E.C. analysed the results and wrote the paper.

## Funding

The Forestry Commission, the University of Grenoble Alpes (through an AGIR project) and Inria (through the Action de Recherche Collaborative PlantScan3D).

## References

Alenyà, G., B. Dellen, S. Foix, and C. Torras (2013). Robotized plant probing: Leaf segmentation utilizing time-of-flight data. *IEEE Robotics and Automation Magazine 20*(3), 50–59.

Bayer, D., S. Seifert, and H. Pretzsch (2013). Structural crown properties of norway spruce (*Picea abies* [l.] karst.) and european beech (*Fagus sylvatica* [l.]) in mixed versus pure stands revealed by terrestrial laser scanning. *Trees 27*, 1035–1047.

Belkin, M. and P. Niyogi (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation 15*, 1373–1396.

Belton, D., S. Moncrieff, and J. Chapman (2013). Processing tree point clouds using gaussian mixture models. In *Proceedings of the ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 43–48.

Bey, A., R. Chaine, R. Marc, G. Thibault, and S. Akkouche (2011). Reconstruction of consistent 3d cad models from point cloud data using a priori cad model. In *Proceedings of the ISPRS Workshop on Laser Scanning*, pp. 289–294.

Casella, E. and H. Sinoquet (2003). A method for describing the canopy architecture of coppice poplar with allometric relationships. *Tree Physiology 23*, 1153–1169.

Chaivivatrakul, S., L. Tang, M. N. Dailey, and A. D. Nakarmi (2014). Automatic morphological trait characterization for corn plants via 3d holographic reconstruction. *Computers and Electronics in Agriculture 109*, 10–123.

Chéné, Y., D. Rousseau, P. Lucidarme, J. Bertheloot, V. Caffier, P. Morel, E. Belin, and F. Chapeau-Blondeau (2012). On the use of depth camera for 3d phenotyping of entire plants. *Computers and Electronics in Agriculture 82*, 122–127.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein (2009). *Introduction to algorithms. 3th ed.* MIT Press.

Côté, J.-F., J.-L. Widlowski, R. A. Fournier, and M. M. Verstraete (2009). The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial lidar. *Remote Sensing of Environment 113*, 1067–1081.

Dassot, M., T. Constant, and M. Fournier (2011). The use of terrestrial lidar technology in forest science: application fields, benefits and challenges. *Annals of Forest Science 68*, 959–974.

Dey, T. K., P. Ranjan, and Y. Wang (2012). Eigen deformation of 3d models. *The Visual Computer 28*, 585–595.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 269–271.

Dornbusch, T., P. Wernecke, and W. Diepenbrock (2007). A method to extract morphological traits of plant organs from 3d point clouds as a database for an architectural plant model. *Ecological Modelling 200*, 119–129.

Douglas, E. S., J. Martel, Z. Li, G. Howe, K. Hewawasam, R. A. Marshall, C. L. Schaaf, T. A. Cook, G. J. Newnham, A. Strahler, and S. Chakrabarti (2015). Finding leaves in the forest: the dual-wavelength echidna lidar. *IEEE Geoscience and Remote Sensing Letters 12*(4), 776–780.

Edelsbrunner, H. (2001). *Geometry and topology for mesh generation.* Cambridge University Press.

Furbank, R. T. and M. Tester (2011). Phenomics – technologies to relieve the phenotyping bottleneck. *Trends in Plant Science 16*(12), 635–644.

Godin, C., E. Costes, and H. Sinoquet (1999). A method for describing plant architecture which integrates topology and geometry. *Annals of Botany 84*, 343–357.

Godin, C. and H. Sinoquet (2005). Functionalstructural plant modelling. *New phytologist 166*(3), 705–708.

Golbach, F., G. Kootstra, S. Damjanovic, G. Otten, and R. van de Zedde (2015). Validation of plant part measurements using a 3d reconstruction method suitable for high-throughput seedling phenotyping. *Machine Vision and Applications*, 1–18.

Haala, N. and M. Kada (2010). An update on automatic 3d building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing 65*, 570–580.

Haines, E. (1989). Essential ray tracing algorithms. In A. Glassner (Ed.), *An introduction to ray tracing*, pp. 33–77. Academic Press.

Hassan, S., F. Hétroy, F. Faure, and O. Palombi (2011). Automatic localization and quantification of intracranial aneurysms. *Computer Analysis of Images and Patterns. Lecture Notes in Computer Science 6854*, 554–562.

Hebert, M. and E. Krotkov (1992). 3-d measurements from imaging laser radars: how good are they? *Image and Vision Computing 10*(3), 170–178.

Hosoi, F., K. Nakabayashi, and K. Omasa (2011). 3-d modeling of tomato canopies using a high-resolution portable scanning lidar for extracting structural information. *Sensors 11*, 2166–2174.

International Plant Phenotyping Network (Accessed: 08-02-2016). http://www.plant-phenotyping.org/.

Kaminuma, E., N. Heida, Y. Tsumoto, N. Yamamoto, N. Goto, N. Okamoto, A. Konagaya, M. Matsui, and T. Toyoda (2004). Automatic quantification of morphological traits via three-dimensional measurement of arabidopsis. *The Plant Journal 38*, 358–365.

Karni, Z. and C. Gotsman (2000). Spectral compression of mesh geometry. In *Proceedings*

*of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pp. 279–286.

Lazarus, F. and A. Verroust (1999). Level set diagrams of polyhedral objects. In *Proceedings of the 5th ACM Symposium on Solid Modeling and Applications (SMA)*, pp. 130–140.

Lévy, B. (2006). Laplace-beltrami eigenfunctions: towards an algorithm that understands geometry. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications (SMI)*, pp. 13.

Li, L., Q. Zhang, and D. Huang (2014). A review of imaging techniques for plant phenotyping. *Sensors 14*, 20078–20111.

Li, Y., X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. Mitra (2011). Globfit: consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics 30*(4), 52.

Lin, Y. (2015). Lidar: An important tool for next-generation phenotyping technology of high potential for plant phenomics? *Computers and Electronics in Agriculture 119*, 61–73.

Livny, Y., F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana (2010). Automatic reconstruction of tree skeletal structures from point clouds. *ACM Transactions on Graphics 29*, 151.

Lou, L., Y. Liu, M. Shen, J. Han, F. Corke, and J. H. Doonan (2015). Estimation of branch angle from 3d point cloud of plants. In *Proceedings of the International Conference on 3D Vision (3DV)*, pp. 554–561.

Nguyen, A. and B. Le (2013). 3d point cloud segmentation: a survey. In *Proceedings of the 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, pp. 225–230.

Paproki, A., X. Sirault, S. Berry, R. Furbank, and J. Fripp (2012). A novel mesh processing based technique for 3d plant analysis. *BMC Plant Biology 12*(1), 63.

Paulus, S., J. Dupuis, A.-K. Mahlein, and H. Kuhlmann (2013). Surface feature based classification of plant organs from 3d laserscanned point clouds for plant phenotyping. *BMC Bioinformatics 14*, 238.

Paulus, S., J. Dupuis, S. Riedel, and H. Kuhlmann (2014). Automated analysis of barley organs using 3d laser scanning: an approach for high throughput phenotyping. *Sensors 14*, 12670–12686.

Paulus, S., H. Schumann, H. Kuhlmann, and J. Léon (2014). High-precision laser scanning system for capturing 3d plant architecture and analysing growth of cereal plants. *Biosystems Engineering 121*, 1–11.

Qiu, H. J. and E. R. Hancock (2007). Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 1873–1890.

Quan, L., P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang (2006). Image-based plant modeling. *ACM Transactions on Graphics 25*(3), 599–604.

Reuter, M., F. E. Wolter, and N. Peinecke (2006). Laplace-beltrami spectra as 'shape-dna' of surfaces and solids. *Computer-Aided Design 38*, 342–366.

Rose, J. C., S. Paulus, and H. Kuhlmann (2015). Accuracy analysis of a multi-view stereo approach for phenotyping of tomato plants at the organ level. *Sensors 15*, 9651–9665.

Rusu, R. B. and S. Cousins (2011). 3d is here: Point cloud library (pcl). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Tao, S., Q. Guo, S. Xu, Y. Su, Y. Li, and F. Wu (2015). A geometric method for wood-leaf separation using terrestrial and simulated lidar data. *Photogrammetric Engineering and Remote Sensing 81*(10), 767–776.

Tilly, N., D. Hoffmeister, H. Liang, Q. Cao, Y. Liu, V. Lenz-Wiedemann, Y. Miao, and G. Bareth (2012). Evaluation of terrestrial laser scanning for rice growth monitoring. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science 39*, B7.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing 17*, 395–416.

Wahabzada, M., S. Paulus, K. Kersting, and A.-K. Mahlein (2015). Automated interpretation of 3d laserscanned point clouds for plant organ segmentation. *BMC Bioinformatics 16*, 248.

Xia, C., L. Wang, B.-K. Chung, and J.-M. Lee (2015). In situ 3d segmentation of individual plant leaves using a rgb-d camera for agricultural automation. *Sensors 15*, 20463–20479.

Xu, H., N. Gossett, and B. Chen (2007). Knowledge and heuristic based modeling of laser-scanned trees. *ACM Transactions on Graphics 26*(4), 19.

Yang, L. (2005). Building connected neighborhood graphs for isometric data embedding. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining (KDD)*, pp. 722–728.

Yin, K., H. Huang, P. Long, A. Gaissinski, M. Gong, and A. Sharf (2015). Full 3d plant reconstruction via intrusive acquisition. *Computer Graphics Forum*.